U2073

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

THE UNIVERSITY OF BIRMINGHAM

Degree of B.Sc.

Artificial Intelligence and Computer Science. Final Examination

Psychology and Artificial Intelligence. Final Examination

Mathematics and Artificial Intelligence. Final Examination

Degree of Combined & General Honours

Arts & Computer Science. Final Examination

SEM3A6

Planning

May 1998 2 hours

[Answer Question **ONE** and **TWO other** Questions]

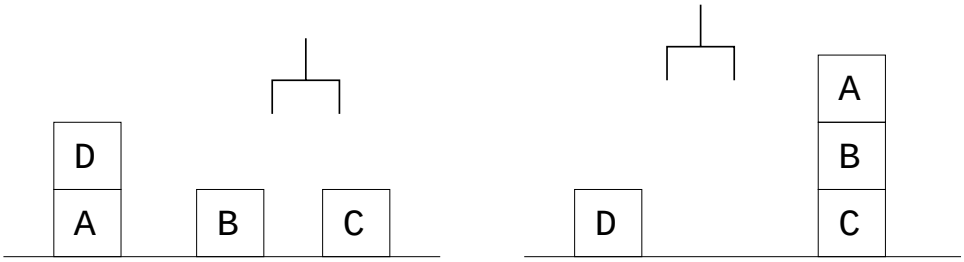
Turn Over

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

[Answer Question **ONE** and **TWO other** Questions]

1. Means-ends analysis and STRIPS

   (a) Let the initial state and goal state below be given. Formalise the problem in STRIPS and show in detail how STRIPS solves this problem. Assume the standard operators of the blocks world, `PICKUP`, `PUTDOWN`, `UNSTACK`, and `STACK`. Assume furthermore optimal choices.



Initial State **Goal State** [20%]

   (b) How does means-ends analysis build up plans? Explain in particular the rôle of backtracking. [7%]

   (c) Is STRIPS complete, that is, does it always find a plan if there is a plan? Explain your answer. [5%]

   (d) Why does the frame problem not occur in STRIPS planning? [8%]

2. Means-ends analysis for simultaneous goals

   (a) What is the Sussman anomaly and why is it called an anomaly? [8%]

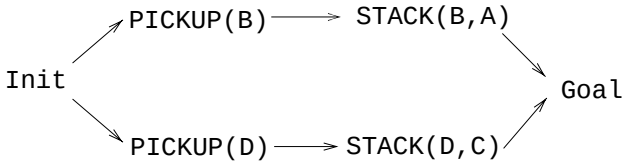
   (b) How is means-ends analysis extended to deal with simultaneous goals? Explain the basic idea and explain in particular the notion "regredient". [12%]

   (c) Discuss advantages and drawbacks of means-ends analysis for simultaneous goals compared to non-linear planning. [10%]

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

3. Non-linear planning

   (a) What different possibilities are there for interpreting a non-linear plan like

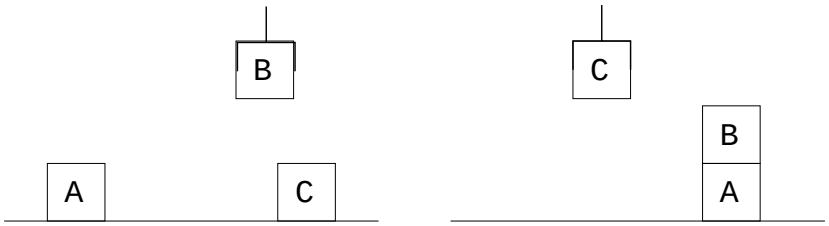


   [6%]

   (b) What is a dependency? [6%]

   (c) What is a parallel conflict and what are the two standard techniques to resolve one? Which of the two is normally preferred? Explain your answer. [6%]

   (d) How is the following planning problem in the blocks world solved by non-linear planning? (Assume the operators `PICKUP`, `PUTDOWN`, `UNSTACK`, and `STACK`.)



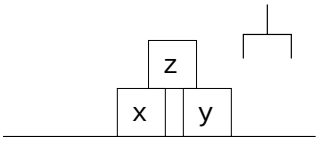
Initial State **Goal State** [12%]

4. Abstraction

   (a) Explain the main idea of situation abstraction. Which order would you choose for the blocks world properties `Clear`, `Handempty`, `Holds`, `On`, and `Ontable`? Explain your choice. [8%]

   (b) Explain the main idea of operator abstraction. In what way are the plan operators dependent on the planning algorithm in operator abstraction? [10%]

   (c) Define an abstract operator `BUILD_ARCH(x, y, z)` that builds an arch from the three blocks `x` (on the table), `y` (in the hand), and `z` (on the table).



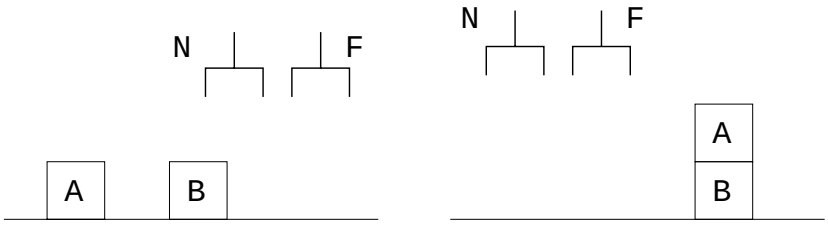
   Assume an additional property `Nearby`. Use the usual operators of the blocks world. Use furthermore the operator `PUTCLOSE(x, y)`, which is similar to the operator `PUTDOWN(x)`, but guarantees in addition that `x` is put near `y`, another block on the table, as well as the operator `STACK_ON_TWO(x, y, z)`, by which a block `z` can be put on two blocks `x` and `y` that are on the table, clear, and nearby each other. [12%]

THIS IS NOT AN OPEN-BOOK EXAMINATION – CANDIDATES MAY NOT CONSULT ANY REFERENCE MATERIAL DURING THE SITTING

5. Conditional plans

(a) How are plans generated that can contain conditionals? [10%]

(b) Assume the following planning situation:



Initial State **Goal State**

There are two hands, one for cheap normal manipulation, N, and one for handling fragile objects, F, which is expensive in execution. Whenever an object is not fragile, prefer hand N over hand F. On plan construction it is not known whether block A is fragile or not. What does a conditional plan for the planning problem look like? [10%]

(c) Discuss the relationship between automatic program construction and conditional planning. [10%]

Answers:

1. Means-ends analysis

   (a) 1st Cycle: Describe the actual state = initial state as:
   On(D, A), Clear(D), Ontable(A), Clear(B), Ontable(B), Clear(C), Ontable(C), Handempty.
   The goal state as:
   Clear(D), Ontable(D), Ontable(C), On(B, C), On(A, B), Clear(A), Handempty.
   Open goals are Ontable(D), On(B, C), On(A, B), Clear(A).
   Select one of the open goals, Ontable(D)
   There is only one operator that fulfils this goal, PUTDOWN(D).
   In a recursive call of the algorithm achieve the preconditions of this operator instance, that is, Holds(D). This can be achieved by UNSTACK(D, A), an operator instance for which all preconditions are met.
   So the first cycle generates the plan fragment P := (UNSTACK(D, A), PUTDOWN(D))
   We have now to update the actual state by simulating the application of the plan fragment to the initial state. We get as actual state:
   Clear(D), Ontable(D), Clear(A), Ontable(A), Clear(B), Ontable(B), Clear(C), Ontable(C).
   2nd cycle: Open goals are On(B, C) and On(A, B)
   Select On(B, C). Only operator instance to achieve STACK(B, C). By recursive call of the planning algorithm the preconditions of the application Clear(C), Holds(B) are met by the plan (PICKUP(B)). So we extend P to
   P := (UNSTACK(D, A), PUTDOWN(D), PICKUP(B), STACK(B, C))
   We get as actual state:
   Clear(D), Ontable(D), Clear(A), Ontable(A), Clear(B), On(B, C), Ontable(C)
   3rd cycle: Analogously to the 2nd cycle we generate the last plan fragment for the only open sub-goal On(A, B), namely (PICKUP(A), STACK(A, B)), hence, we get as plan in total:
   P := (UNSTACK(D, A), PUTDOWN(D), PICKUP(B), STACK(B, C), PICKUP(A), STACK(A, B))
   Updating the actual state, we get goal state, hence planning terminates.

   (b) Means-ends analysis decomposes the planning problem into sub-problems and attacks the different sub-goals independently by working backward from the sub-problem to the initial state. It heuristically selects an operator that has the sub-goal in the add-list and recursively generates a plan that makes this operator applicable. When a plan for one sub-goal is found the actual state of the world is updated and a plan for the next sub-goal is generated until all sub-goals are satisfied. Newly generated plan fragments are always appended to the end of the plan.
   There are two choices, the selection of a sub-goal and the selection of an operator instance to fulfil that sub-goal. If further planning fails, backtracking to the most recent choice takes place.

   (c) No, it is not. If STRIPS has to plan for swaping the values of two variables it

too narrowly concentrates on one sub-goal and destroys the other value. That is, although there exists a plan that solves this problem (using an auxiliary variable) STRIPS is not able to find it.

(d) In STRIPS there is a search in the state-space. Calculations take place in one particular model and not in all possible models.

2. Means-ends analysis for simultaneous goals

(a) The Sussman anomaly is a blocks world problem with three blocks for which the means-ends strategy can't find an optimal solution independently of the heuristics used. It is called an anomaly since one assumed that it would be possible to create optimal plan by optimal heuristics.

(b) Means-ends analysis is extended by introducing an additional choice, namely the newly generated plan is not just appended to the old plan, but inserted somewhere in the old plan. This is achieved by partitioning the old plan P in sub-plans P' and P''. Furthermore goals one plans for are marked as protected and all partitions that lead to a plan that destroys protected goals are rejected. Since P'' is executed after the new plan, one has to calculate what has to hold before the execution of P'' such that the selected sub-goal holds after the execution of P''. The weakest condition that guarantees this is called regredient.

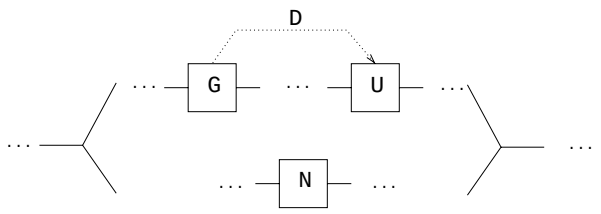
(c) Advantages of means-ends analysis for simulatneous goals are: It is still much simpler than non-linear planning, there is an actual state and a focus of planning. Advantages of non-linear planning: Greater flexibility in execution, potentially less backtracking since no choice of partitioning necessary.

3. Non-linear planning

(a) Essentially there are three different interpretations: respecting the order whole branches can be executed in arbitrary order (weak non-linearity), operators in parallel branches can be executed in arbitrary order as long as the order is respected (strong non-linearity), operators in parallel branches can be executed in parallel (concurrency).

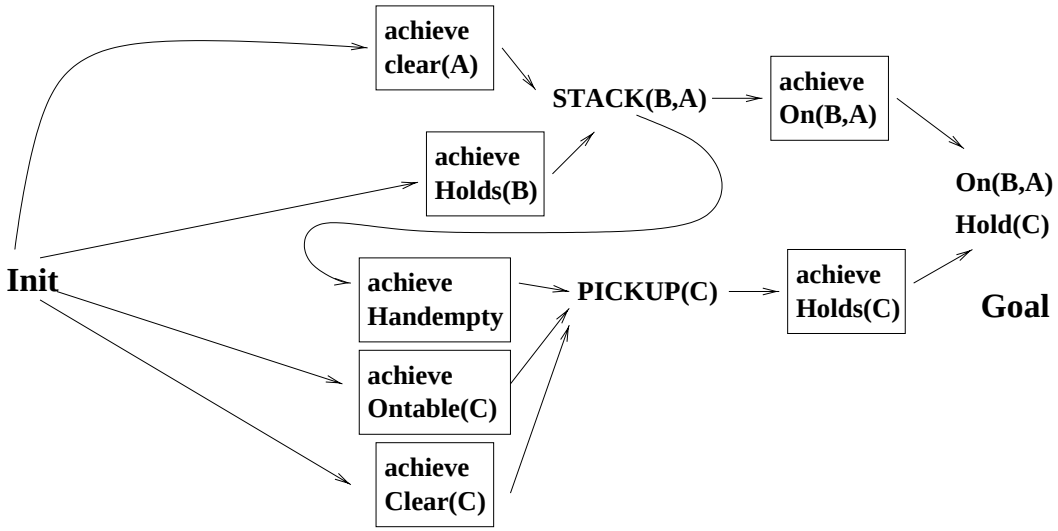
(b) A dependency is the relation between a node G that generates a property prop, a node U that uses prop in a plan P, such that G comes before U and no node between G and U generates prop.

(c)



A parallel conflict is a situation in which a node `N` that uses a property `prop` can be inserted in a dependency between the generator `G` and the user `U` of that very property.
A parallel conflict can be resolved by further linearisation (normally preferred in order to keep the plan small) or by adding additional nodes to the plan.

(d)



4. Abstraction

(a) The idea of situation abstraction is to initially neglect less important properties and recursively refine an abstract plan by considering more and more less important properties until eventually all properties are considered.
In decreasing importance: `Ontable` > `On` > `Clear` > `Holds` > `Handempty`. `Ontable` is most important since it has not to be changed again. `On` comes second, since primarily the blocks world is about building towers. `Handempty` comes last, since it changes in each action.

(b) In operator abstraction, one operator stands for the execution of different operators, represented in the so-called plot. The plot is a plan fragment that replaces the abstract operator. Since an abstract operator contains plan fragments it depends on the plan data structure of the planning algorithm.

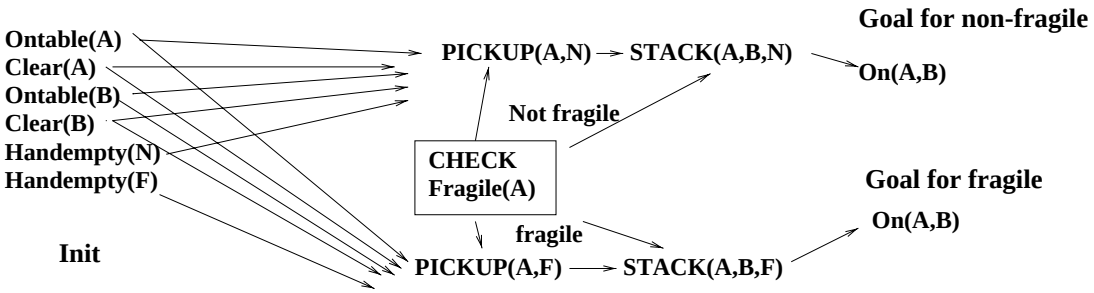
(c) BUILD_ARCH(x, y, z)

preconditions	Holds(x)
	Ontable(y)
	Clear(y)
	Ontable(z)
	Clear(z)
delete_list	Holds(x)
	Clear(y)
add_list	Ontable(x)
	Nearby(x, y)
	On(z, x)
	On(z, y)
plot	(PUTCLOSE(x, y), PICKUP(z), STACK_ON_TWO(x, y, z))

5. Conditional plans

(a) If certain facts are not known during plan generation, but finitely many possibilities exist, it is possible to generate for each of the possibilities a plan. The information is acquired in plan execution and according to the case at hand the corresponding conditional part is executed.

(b)



(c) If in addition to conditional there are constructs for loops in plans, the essential ingredients of a Turing complete programming language are given and plan construction and program construction are equivalent.